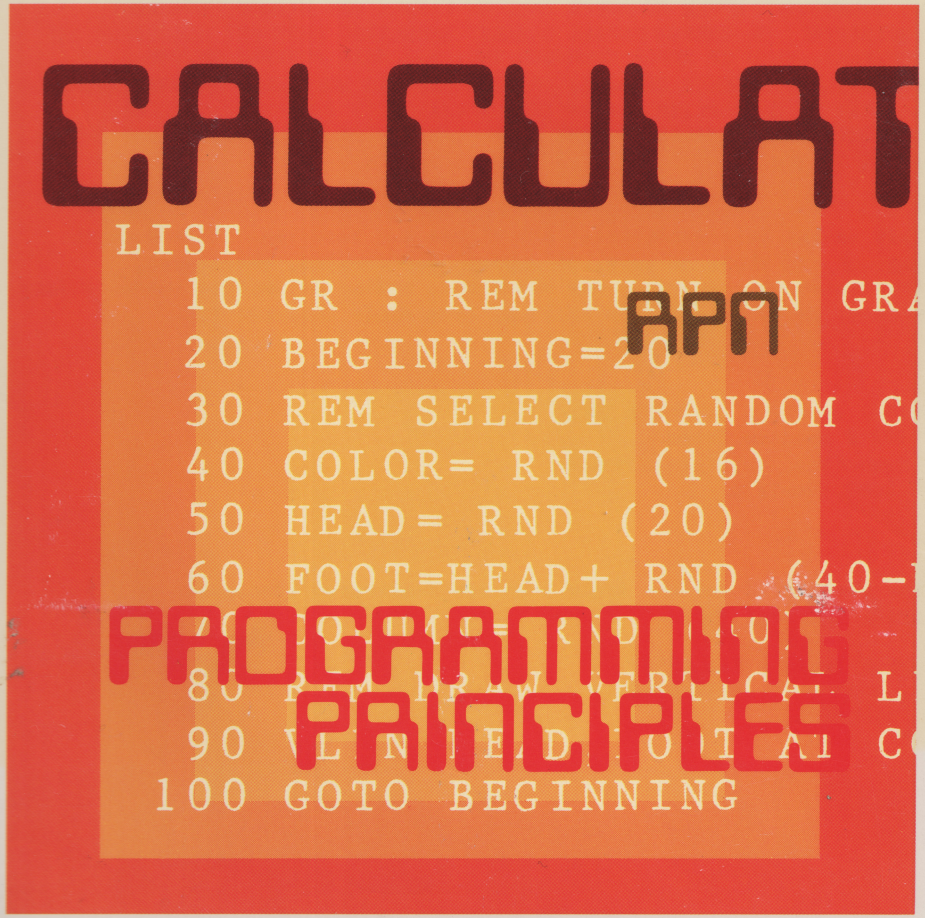
Apple II

Apple How To!

Lessons in Computing and Calculating



NOTICE

Apple Computer Inc. reserves the right to make improvements in the product described in this manual at any time and without notice.

DISCLAIMER OF ALL WARRANTIES AND LIABILITY

APPLE COMPUTER INC. MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL OR WITH RESPECT TO THE SOFTWARE DESCRIBED IN THIS MANUAL, ITS QUALITY, PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. APPLE COMPUTER INC. SOFTWARE IS SOLD OR LICENSED "AS IS". THE ENTIRE RISK AS TO ITS QUALITY AND PERFORMANCE IS WITH THE BUYER. SHOULD THE PROGRAMS PROVE DEFECTIVE FOLLOWING THEIR PURCHASE, THE BUYER (AND NOT APPLE COMPUTER INC., ITS DISTRIBUTOR, OR ITS RETAILER) ASSUMES THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR, OR CORRECTION AND ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES. IN NO EVENT WILL APPLE COMPUTER INC. BE LIABLE FOR DIRECT, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT IN THE SOFTWARE, EVEN IF APPLE COMPUTER INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES OR LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.

1Ø26Ø Bandley Drive
Cupertino, California 95Ø14
(4Ø8) 996-1Ø1Ø

APPLE Product #A2LØØ5Ø
(Ø3Ø-Ø14Ø-A)

Apple II

Apple How To!

Lessons in Computing and Calculating

PREFACE

This manual accompanies the APPLE HOW TO! diskette. The first chapter of the manual gives you an overview of each of the programs on the APPLE HOW TO! diskette and tells you how to use this diskette with your particular Apple configuration. Each of the remaining chapters gives you detailed information about each of the APPLE HOW TO! programs.

The pointing hand symbol appears throughout this manual. Its purpose is to indicate something particularly interesting or useful.

TABLE OF CONTENTS

CHAPTER 5

ASSEMBLE-IT-YOURSELF MULTI-TONE KIT 45

INDEX 65

CHAPTER 1
INTRODUCTION, OR, HOW TO USE APPLE HOW TO!

Welcome to APPLE HOW TO! The set of programs that make up this portion of Apple's educational software series will increase both your knowledge of computers and your ability to wow your family and friends with fancy programming tricks. APPLE HOW TO! is composed of four programs: the RPN Calculator, a program that turns your Apple into a keystroke calculator based on RPN, a highly logical system for doing simple or complex calculations; Rod's Color Pattern, a carefully-documented program that helps you understand how BASIC works; Scrolling Window Tutor, a tutorial program that teaches you how to create text windows ("What is a text window?" you ask. Select the Scrolling Window Tutor and find out!); and the Assemble-It-Yourself Multi-Tone Kit, a program that creates some pretty bizarre sounds while teaching you the fundamentals of assembly-language programming.

The lessons in APPLE HOW TO! will be challenging and interesting to anyone of junior high school age or older.

APPLE HOW TO! works with any Apple II or II Plus that has at least 48K of memory and firmware versions of both Applesoft BASIC and Integer BASIC. If you have an Apple II Plus and you want to use APPLE HOW TO! you must have a plug-in Integer BASIC firmware card or the Apple Language System. Similarly, if you have an Apple II and you want to use APPLE HOW TO! you must have a plug-in firmware Applesoft card or the Apple Language System. In addition, you need at least one disk drive set up to use 16 sector diskettes, and a video monitor. If you have a color monitor, use it: Rod's Color Pattern, one of the programs included in APPLE HOW TO! is a lot more fun in color.

Your APPLE HOW TO! programs are stored on a 16 sector diskette. If you use DOS 3.2 (or a DOS with a lower version number) your disk drive is set up for use with 13 sector diskettes, and your Disk II interface card must be modified to use 16 sector diskettes. Your dealer can sell you a DOS update package that will help you update your DOS to the latest available version, and update any old 13 sector diskettes you may have.

CHAPTER 2
RPN CALCULATOR

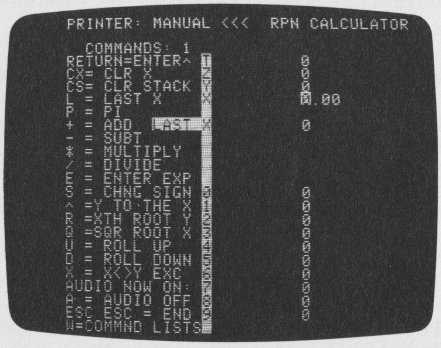
Program Language: Applesoft BASIC

The Apple RPN Calculator is a full-blown scientific keystroke calculator with a four-register stack, ten addressable memory registers, and a multiplicity of functions. Its design follows that of traditional RPN calculators, making it an excellent tool for the teaching of RPN techniques. The Apple RPN Calculator has an enormous advantage over other RPN calculators in that it allows you to see all the stack and memory registers simultaneously.

If you haven't used RPN logic before, it may seem stilted and strange at first. But, after you have worked through a number of problems, you will discover how natural a system it really is.

GETTING STARTED

Select 1 RPN CALCULATOR from the main diskette menu. After you have passed the title page by pressing the space bar as directed, you will be presented with the calculator itself.



When you exit from the calculator, the mode in which it is operating is stored, so that next time you use the calculator, you can pick up right where you left off. One of the things that is stored is whether you are displaying command lists on the left side of the screen, as illustrated above. If you do not see the list shown, please press the W key.

RPN is a keystroke calculator, meaning that rather than pressing a number of keys simultaneously, you just press single keys (or sometimes a sequence of single keys) to get things to happen, just as you would on a typical calculator. Under the heading COMMANDS: 1, the left side of the screen shows a list of commands and abbreviated descriptions of what these commands do. For example, pressing the + key will cause two numbers to be added together, and pressing the S key will change the sign of a number.

To the right of the command list, there's a column of letters and numbers on a vertical white stripe. These letters and numbers are the names of the memory registers. ("Register" is a fancy word for memory location.) The four registers at the top, T, Z, Y and X, make up the "stack," a subject we will spend a good deal of time discussing later. Below the stack is a special register called LAST X, which allows you to recover from typing errors and other mistakes. The ten numbered registers below the stack are ten memory registers in which you can store numbers.

On the right side of the screen is a column of zeros. Notice that each Ø in this column corresponds to one of the registers in the white column. The Ø that corresponds to the X register has a flashing cursor over it and shows zeros in the two decimal places to the right of it. You will find out why the X register is special later on.

The very top line of the screen says

```
PRINTER: MANUAL <<<  RPN CALCULATOR  >>>
```

For now you can ignore this line. It will be discussed later in the section called "Printing."

The last command on this menu is W=COMMAND LISTS. Press the W key now to display the second command list. The second list of commands is called COMMANDS: 2, and deals mostly with functions that affect the calculator's memory. Each of the memory commands begin with an M.

Press the W key again to go to the third command list, COMMANDS: 3. The COMMANDS: 3 list consists primarily of commands that cause the calculator to perform scientific functions. Each of these function commands begins with an F.

Again press the W key to display the final command list, COMMANDS: 4. COMMANDS: 4 consists of printing commands. These commands require that you have a printer attached to your Apple.

The line ESC ESC = END is displayed near the bottom of each command list. Pressing the ESC key twice, as indicated by this line, will end the program, store the contents of the memory registers, and return you to the main diskette menu. To return to the COMMANDS: 1 list from COMMANDS: 4, press the W key once again.

These lists represent all the commands you will use with the RPN Calculator program. In the following sections we will go through each operation. If you have never used RPN math, you are in for an interesting adventure. If you have, you will find it a treat to work on a "transparent" calculator.

THE ADVANTAGES OF RPN

RPN calculators work using arithmetic notation, the most fundamental and easy-to-use method of working math problems. You used arithmetic notation when you were in grammar school, before you were introduced to the equals sign (algebraic notation). Arithmetic notation looks like this:

```
  2
+ 2
---
```

As you will see when you begin working with the calculator, this is very similar to the way you will enter problems, except that you will say

```
2
2 +
---
```

It really is that easy. And what are the advantages? Many. One problem with algebraic notation, such as

```
(27 + 3) * ((14 + 4) / (33 + 2)) =
```

is that you can get lost in a sea of parentheses. There are eight in this problem, and moving one of them to the wrong place makes the answer come out differently. RPN has no parentheses. You don't need them. You are never working with more than one simple calculation at a time. This system automatically serves to simplify problem solving.

Unlike calculators based on algebraic notation, an RPN calculator automatically holds intermediate results in the stack registers, where you can view them at any time. As you will see, the whole process of problem solving is very natural. When you need an intermediate result, such as the result of (27 + 3) in the example above, RPN will answer that portion of the problem, and then await your next move.

ENTERING NUMBERS

When using the Apple RPN Calculator, you enter all numbers into the X register. This is the one with the flashing cursor on it. It is the

heart of the calculator, and practically all your work is done here. It is the only register you would normally see on a regular calculator without using special commands. To enter a number, just type in the number as you would with any calculator, using a decimal point if the number has a fractional part. Try typing the number 546.829.

CHANGING SIGNS

To make a positive number negative, or vice versa, press the S key for Sign. Change the number you just typed to -546.829 by pressing the S key. If you press it twice, the number will become positive again. (The + sign is not shown; if there is no sign, the number is assumed to be positive.)

CLEARING THE X REGISTER

You can clear whatever you have entered in the X register by pressing the C key for Clear and the X key to specify the X register. You can then either re-enter the number or enter a new number. The arrow keys are not active in RPN; if you make a mistake, you must retype the entire number, just as with a regular calculator. (Sorry about that.)

Clear the X register now so that you can begin solving problems. (Ignore the flashing X for now. We will discuss its meaning later on.)

PROBLEM SOLVING IN RPN

Let's start out with a simple problem:

```
  2
  3 +
-----
```

If you were working this out on paper, you would

1) write down the 2
2) write down the 3
3) add them together

This is exactly what you will do with the RPN Calculator.

First, type a 2 into the calculator's X register. (Just press the 2 key.) Then officially ENTER^ the number by pressing the RETURN key. The 2 is now displayed twice: in the X register as 2.ØØ and in the Y register as 2. In the Y register, it is safe from harm, but in the X register it is perishable: if you attempt to enter another number, the 2.ØØ in the X register will disappear. You can tell that the number in the X register is perishable because the X is flashing.

From now on, when you are instructed to ENTER^ a number, press the RETURN key. The ^ in ENTER^ indicates that the number entered will be pushed up in the stack.

Now press the 3 key to put the 3 in the X register. You will see that the 2.ØØ in the X register is replaced by the 3, but the 2 in the Y register is unaffected. Also note that the cursor is to the right of the 3. This indicates that you can continue adding digits to the number if you wish. In this case, however, we do not want to enter any more digits.

Now that both numbers are displayed on the screen it is time to add them together. To do so, press the + key. (It is unnecessary to use the SHIFT key when using the RPN Calculator on the Apple II. In this case, just press the semicolon/plus key. The program accepts this as a plus. If you do use the shift key, however, the operation will work equally well.)

The 2 and the 3 have been added together and the result has been placed in the X register. The 3 that was in the X register has now been moved to the LAST X register, which always displays the number that was last in the X register before you performed some function. The purposes of the LAST X register will be discussed later.

A MORE COMPLICATED PROBLEM

Try this one:

(4 + 9) * (6 - 2)

If you were to work this out on paper, you would

1) Add 4 + 9 and write down the answer.
2) Subtract 2 from 6 and write down the answer.
3) Multiply the first answer by the second answer.

This is exactly what you will do with this problem. Begin by clearing the X register (CX). This step is not really necessary, as the calculator will keep old things from getting in your way, but it is nice to start with a clear field when you are learning.

After clearing the X register, do the following:

1) Type a 4 and ENTER^.
2) Type the number 9.
3) Press the + key.

At this point the screen is displaying a 9 in the LAST X register and 13.ØØ in the X register. The cursor lies on the 3 in 13.ØØ. The X register is now in a third state. (We have already discussed two states of the X register. In the first, when the X is flashing, the entire number currently in the X register disappears from the X register when you type a digit, and the new digit begins a new number. In the second, the cursor to the right of the number indicates that the number is "in progress" and any digits you type will be added to those already on the screen.) The new state is this: the cursor is

on the "1's" column and the X is not flashing. Now, when a new digit is entered, the old number is pushed up on the stack. The number that is now in the X register moves up to the Y register. As you will soon see, whatever is in the Y register will, in turn, be pushed up to the Z register, etc., with the contents of the T (Top) register being pushed out and lost.

Continuing with the example, we must next compute 6 - 2. To do so, begin by typing a 6. You will see that the 13 is pushed up to the Y register and the 6 is placed in the X register. Now ENTER^ the 6, and then type a 2. The stack, from top to bottom, will now read Ø, 13, 6, 2. Press the - key to subtract. The 2 is subtracted from the 6, and the resulting 4 is placed in the X register. The 13 drops down from the Z register into the Y register, ready for the next step.

The next step is to multiply the two answers together. And here they are, just waiting for you to press the * key! Press the * key now to get the final answer, 52.

REVIEW

Here is a summary of the concepts that have been covered so far:

* All numbers are entered into the X register.
* All functions occur immediately when you press the appropriate key: when you press +, addition occurs at that time; you do not have to press the = key.
* Problems are solved in the same order you would solve them if you were using pencil and paper.
* The position of the cursor indicates what will happen to the digits you type:
    a) If the cursor is to the right of the number, the number is "in progress."
    b) If the cursor is on the 1's column and the X is flashing, the contents of the X register is scheduled to be destroyed.
    c) If the cursor is on the 1's column and the X is not flashing the next digit typed will "push the stack up": everything will move up one step.

Now try the following problem:

$$\frac{(2 + 7) * (9 - 4)}{(7 + 5)}$$

KEYSTROKES	RESULTS IN X REGISTER
2 ENTER^ 7 +	9
9 ENTER^ 4 -	5
*	45
7 ENTER^ 5 +	12
/	3.75

Here are some problems to work out by yourself:

$$\frac{23 * (14 + 3)}{68 / 2}$$

((12 + 6) * 2) * (17 + 4)

13 / 4 + (5 + (6 / 4))

In each case, work from the inside out: the 14 + 3 before the 23, the 12 + 6 before multiplying by 2, etc. Take your time and compare your answers with the results (and the number of manipulations you must go through) on your regular algebraic calculator.

MORE ON THE STACK

The stack, as you have seen, consists of the X, Y, Z, and T registers. Stacks are a regular part of all RPN calculators, but usually remain unseen by the user.

There are two types of stacks: FIFO, or First-In-First-Out, and LIFO, or Last-In-First-Out. Basically, a FIFO stack is set up so you put things in at the top and take them out the bottom, while a LIFO stack is set up so you push things in at the bottom and pull them back out from the bottom. RPN calculators use a LIFO stack. The X register is the only entrance to, and exit from, the stack.

CLEARING THE STACK

When you have finished your calculations, you will probably have all sorts of things hanging around in the registers. While they will do no harm, you can clear the entire stack by typing

CS

for Clear Stack. You will then be left with all Ø's (except in the LAST X register, which is not officially part of the stack).

MOVING STACK REGISTERS

Once you have some numbers in the stack, it can be useful to move them around. For example, you may want to add the result you got two computations ago to a new number. However, the result is now up in the T register, and you don't want to lose what is in the Y and Z registers to get to it. Fortunately, there are commands for moving

the stack: U for up and D for down. To see how they work, first fill the stack by typing

```
4.44 ENTER^
3.33 ENTER^
2.22 ENTER^
1.11 (Do not ENTER^)
```

Remember that when you ENTER^ numbers into the stack, the contents of the X, Y, and Z stacks are pushed up, but whatever is in the T register is pushed right off the top of the stack. The Rotate Up command (U), however, takes the contents of the T register and puts them in the X register. Try pressing the U key several times to see what happens. Then press the D key to see the numbers rotate the other way. (Note that the cursor is over the 1's column, and the X is not flashing. The next number you enter will push the stack up one position. Rotating the stack either up or down terminates digit entry, and sets the stack to move up with the entry of a new digit.)

End your experiments with 4.44 in the T register and 1.11 in the X register, the way you entered them originally.

A command that you will probably use more often is the eXchange X and Y command, (the X key). If you press it, you will see that the 2.22 and 1.11 are swapped. Press it again, and they return to their original positions. This key is often used when you want to subtract the contents of Y from that of X, divide the contents of X by that of Y, or perform some other operation where the contents of the two registers must be switched in order to carry out the operation.

As you use the RPN Calculator and your skill and speed increase, you will find that you seldom need to manipulate the position of the numbers being held in the stack. The numbers will almost always be just where you need them, when you need them. But, at the beginning, you may find that you have the wrong result in the wrong place at the wrong time. With these three commands, you can resequence the stack contents into any order.

THE SOUND OF MATH

You can control the sound of the calculator with two keys: A and K. Pressing A turns the Audio -- the prompting and warning sounds of the calculator -- alternately on and off. Once the main audio has been turned off, you can additionally turn off the key click tone by pressing K. As your speed on the calculator increases, you may find that you sometimes press the keys faster than the calculator can accept them. The key click lets you know, without your looking at the screen, that the calculator has accepted each character you type.

FUNCTIONS AND THE STACK

The functions the RPN Calculator performs can be divided into two categories: those using one number and those using two numbers. So

far we have been doing two-number functions, such as addition, subtraction, multiplication, and division. The one-number functions, such as Square Root of X and Reciprocal of X, affect only the contents of the X register; the other three stack registers remain unchanged. For example, ENTER^

```
40
30
20
```

and then type a

```
9
```

followed by a

```
Q
```

The Square Root key, Q, gives the square root of the number 9, leaving the stack like this:

```
T          40
Z          30
Y          20
X           3.00
```

As you can see, the Y, Z, and T registers were not affected when you pressed the Q key.

The effect of two-number operations on the stack is a little more extensive. Subtract the 3 in the X register from the 20 in the Y register by pressing the - key. The stack now contains the following numbers:

```
T          40
Z          40
Y          30
X          17
```

You can see that the result of the latest calculation, 17, now appears in the X register, and the entire stack was dropped one position. Notice that the 40 that originally appeared in the T register now appears both in its original position in the T register and in the Z register to which it was dropped.

Again subtract the contents of the X register from the contents of the Y register by pressing the - key. The result of your calculation again appears in the X register, and now the 40 appears in the T, Z, and Y registers.

This feature is particularly useful when you are doing a series of calculations in which one number remains constant. For example, assume you run a retail store and have received a batch of merchandise. The price of each item you received must be multiplied

by your store mark-up. If you sell at 23% above your cost, you will wish to multiply each item by 1.23 (your cost, 1, plus 23% or .23). To set the stack up for this maneuver, type 1.23 and then ENTER^ 3 times. The stack will look like this:

```
T       1.23
Z       1.23
Y       1.23
X       1.23
```

Let's say the costs of the items are $2.53, $14.95, $38.12, and $7.56. First type 2.53 and then press the * key. The result is displayed in the X register, and you can write it down. Now clear the X register with CX and go through the same procedure with the next amount, 14.95. Continue in this manner until all the items have been marked up.

When you are doing chain calculations, the lifting and dropping of the stack keep intermediate results just where you want them, ready for the next calculation. Try doing such chain calculations as

```
28 + 12 + 35 - 17 + 128 - 14 + 3
```

and you will see that the subtotal always remains in the X register waiting to be lifted into the Y register when the next number is entered. You don't have to keep track of anything.

THE PANIC REGISTER: LAST X

LAST X always holds the number that was in the X register just before the last computation was made. You can recover that number by pressing L, the LAST X key. Let's say you wish to compute the retail price of an item which costs you $187.5Ø. You need to multiply the number by your mark-up of 1.56, but you mistakenly enter 1.65. Type

```
187.5Ø ENTER^
1.65 *
```

The wrong answer, 3Ø9.38, ends up in the X register, and the mistakenly typed mark-up, 1.65, remains in the LAST X register. To reverse the last calculation so it can be redone correctly, simply divide the incorrect answer in the X register by the contents of the LAST X register. Just type

```
L /
```

This puts 187.5Ø back in the X register. Now you can do the calculation again, this time with the correct mark-up. Type

```
1.56 *
```

The item's correct retail price, $292.5Ø, should now be in the X register, and the LAST X register should contain the multiplier, 1.56.

If you now want to multiply the cost of another item, say $23.3Ø, by your standard mark-up of $1.56, type

```
23.3Ø
L
*
```

Your answer will be $36.35. You can continue entering new numbers and multiplying them by your constant, $1.56, in this manner. In fact, you can carry out any two-number function using the constant held in the LAST X register.

NUMBERS, LIMITS, AND DISPLAYS

Try typing

```
1.257
```

but don't ENTER^ it yet. The number appears in the X register just as you typed it. Now ENTER^ the number. The number has been moved to the Y register, and a new number, 1.26, is now in the X register. If you now press RETURN to ENTER^ the contents of the X register should move up to the Y register, right? Try it.

The Y register shows the "raw" number, 1.257, while the X register displays the rounded-off value, 1.26. Actually, only the display has been rounded. The number in the X register remains 1.257. You can test this by rotating the stack up once to move the contents of the X register into the Y register. Press the U and D keys once to try this.

You can change the number of decimal places displayed in the X register with the Fix Display command. Any number of decimal places from zero to eight is allowed. To set the display to 8, type

```
F D 8
```

The X register will now show eight digits to the left of the decimal point. Please return to two digit display by typing

```
F D 2
```

SCIENTIFIC NOTATION

You can enter numbers as large as 999999999 and as small as .Ø1, just by typing them, but if you wish to use numbers outside that range you must use scientific notation.

The full range of numbers that can be handled by the calculator is from -8.5 * 1Ø ^ 37 to 8.5 * 1Ø ^ 37. Any calculation outside this range will cause an OVERFLOW ERROR message, and leave the calculation unfinished.

Let's say you want to use the number, 1.2 * 1Ø ^ 15. First type

1.2

and then press the Enter Exponent key

E

Two zeros will appear near the right edge of the screen, followed by the flashing cursor. This indicates that the calculator is prepared to receive the exponent of 1Ø that you specify. Type

15

If you wanted to change the exponent or correct a typing error, you could just type the new number, displacing the 15.

When the exponent is the way you want it, ENTER^ the number. The display will now read

1.2E+15

This is the way Applesoft normally displays 1.2 * 1Ø ^ +15.

The Enter Exponent function only works with exponents of 1Ø. To use exponents of other numbers, see the section called Exponents.

NEGATIVE EXPONENTS

Negative exponents are entered in almost exactly the same way. The difference is that the S key must be used to change the sign of the exponent. This can be done any time after the E key has been pressed and before the exponent has been entered. For example, to enter 4.3 * 1Ø ^ -11 you could type

4.3
E
1
S
1

and then ENTER^.

If you try to use an exponent that will make your number too large or too small, the exponent will not be accepted. Try entering

1345 * 1Ø ^ -45

The calculator will not accept it and will instruct you that the largest exponent you can enter in this case is 34. Because the size of the number depends on the number of digits to the left of the decimal point, the highest allowed value for the exponent will vary.

If you now type

22

and then ENTER^, the number will be accepted. The new number is now displayed as 1.345E-19. Numbers displayed in scientific notation will always have 1 digit to the left of the decimal place, with as many as 8 digits to the right.

PRECISION

In general, the RPN Calculator is precise to at least eight decimal places, more often nine. This calculator differs from the pocket and desk-top calculators on the market in that it does all its math in base 2, and the translation of numbers from decimal, (base 1Ø) to binary (base 2) and back again occasionally produces slight errors. Just as decimal calculator math has problems dealing with certain numbers (1Ø / 3 = 3.333333333..., but 3.333333333... * 3 = 9.999999999...) binary arithmetic has problems with some of the "easiest" calculations. For example, 1Ø ^ 5 - 1ØØØØØ would leave a remainder if binary arithmetic were being used! The RPN Calculator does some error trapping to alleviate most such obvious problems so it allows you to divide 1Ø by 3 and multiply the result by 3 to get 1Ø back again. Nevertheless, the precision of the calculator does depend somewhat on the numbers used in your calculation. The relationship of certain numbers to both base 1Ø and base 2 is such that the precision can slip to as low as eight places, or one part in 1Ø million.

Because of this "wobbling" precision, the normal Applesoft display of nine significant digits is not allowed by the formatter. You can see all nine, of course, when the number is being displayed in scientific notation since the formatter is then turned off.

You can have faith in the first seven digits of the display and generally, even if you have chained together a good many calculations with long numbers to arrive at your final result, the eighth digit will be accurate within + or - 1. This level of accuracy should prove more than sufficient for most applications. In a calculation of the length of the Golden Gate Bridge (645Ø feet), the answer would be off by less than 1 thousandth of an inch--less than the thickness of this page.

THE MEMORY REGISTERS

There are ten addressable memory registers in the RPN calculator. The memory registers are located just below the stack and are numbered from Ø to 9. You can store and retrieve numbers in any of these memories. You can also do arithmetic operations directly upon values in any or all memory registers, as explained below. The COMMANDS: 2 list displays the options that deal with the calculator's memory registers. Press the W key to display these.

Any time that one of the command lists is displayed, typing M for memory automatically displays the COMMANDS: 2 list. After the memory operation has been completed, the original command list will be redisplayed. But since it takes a certain amount of time to switch these lists, the program will run slighly faster if you leave COMMANDS: 2 on the screen when doing a lot of work with the memory registers.

As explained below, descriptive labels for the memories can be displayed instead of command lists. When these labels are showing, the COMMANDS: 2 list will not normally be displayed. You can temporarily display it by typing

```
M W
```

and then proceeding to enter the rest of your memory command, or pressing the space bar to cancel the operation.

In the following list, the question marks indicate you can enter a number from Ø to 9 to choose which of the memories you wish to affect, or, as appropriate, an A for All, a D for Diskette, or an L for Label.

The ten commands that affect the memories are:

* MC?- clears memories to zero
* MS?- stores a number from the X register to a memory register or the whole memory set to diskette
* MR?- recalls a previously stored number from a memory register or a memory set from diskette
* MX?- exchanges the numbers in the X register with a memory register or the whole memory set with a memory set stored on the diskette
* MSL- stores your current set of descriptive labels on diskette
* MRL- recalls from diskette a set of descriptive labels
* MT?- allows you to enter or change a label or set of labels
* MTT- allows you to give a title to a set of memory labels
* MVL- lets you view the descriptive labels for the memory registers
* MVD- lets you view the titles of each memory set on the diskette

SIMPLE MEMORY STORE

Let's say you are building a four-room house and want to buy the eight foot high two-by-four studs that support the walls. The living room requires 7Ø studs for the wall; the master bedroom, 5Ø studs; the second bedroom, 4Ø studs; the kitchen, 4Ø studs; and another 4Ø studs are needed for the bathroom and entranceway. How many studs are needed, and how much will the total cost of the studs be at $2.85 each?

The first problem is to add up the total number of studs. Type

7Ø ENTER^ 5Ø + 4Ø + 4Ø + 4Ø +

which gives a total of 24Ø. (You could have used the LAST X register to add the last 2 numbers by typing

7Ø ENTER^ 5Ø + 4Ø + L + L +

instead.) Let us store this first answer, the total number of studs needed, in memory register Ø. To do so, type

M S Ø

As soon as you press the M key, the cursor drops down just below the memory registers, and the center of the display announces that you are working with the memories. Then, when you press the S key, the word "STORE" appears, confirming that you have the right function. If you have not yet entered the memory number and you change your mind or press the wrong key, pressing the space bar will cancel the memory operation.

When you type the memory register number, in this case Ø, the contents of the X register will be duplicated in the Ø memory register. The number in the X register will remain there.

To get the final answer, multiply the 24Ø studs by the cost of $2.85 per stud by typing

2.85 *

The result of this operation is $684.ØØ .

(If your display is still fixed at eight decimal places, you can return to using two by typing

F D 2

since you are dealing with money.)

Now store this result in memory register 2 by typing

M S 2

RECALLING FROM A MEMORY

It might be interesting to know how many board-feet of lumber you are buying. To do so, first recall the number of studs and then multiply that figure by 8ft. To recall the total number of studs stored in memory register Ø, type

M R Ø

This operation duplicates the contents of memory register Ø, namely the number 24Ø, in the X register. It does not affect the 24Ø held in memory register Ø, which remains intact. Multiplying the number by 8 is left up to you. As you can tell by the location of the cursor over the 1's column and the absence of a flashing X, you can simply proceed to enter the 8 and multiply by it. When you have finished, store the result in memory register 1.

MEMORY EXCHANGE

Suppose that you have done some more checking and have found a place that sells 16 foot two-by-fours for $.34 a foot. Let's find out how much money, if any, you would save if you bought 16 footers and cut them yourself. Multiply the number of board-feet, the result of your last calculation, by .34. The result, $652.8Ø, is an obvious savings over the previous price.

In order to subtract one price from the other, you need one in the Y register and the other in the X register. Let's first duplicate the contents of the X register by pressing the ENTER^ key. The contents of the registers in question will now look like this:

```
Y          652.8
X          652.8Ø

Ø          24Ø
1         192Ø
2          684
```

Now exchange the contents of the X register with the contents of memory 2 by typing

```
M X 2
```

The new lower price is stored in memory register 2, and the registers now look like this:

```
Y          652.8
X          684.ØØ

Ø          24Ø
1         192Ø
2          652.8
```

The only problem is that if you try to subtract now you will be subtracting the larger number from the smaller number, which is exactly the wrong way around. So, let's exchange the X and Y registers by pressing the X key.

```
Y          684
X          652.8Ø
```

Now press the - key, and the $31.2Ø you save will appear in the X register.

If the foregoing seemed difficult and complex, give yourself time. It takes a while to get used to new things. The RPN Calculator is a very powerful beast and an unusual one. With a few hours of use, it will begin to seem very natural indeed. And if you look at the above procedure, you will see that there were really very few keystrokes involved. They were just very new to you.

CLEARING A MEMORY

Here's one that's easy. The number 192Ø is no longer needed, and we wish to clear it. Just type

M C 1

and as the screen announces MEMORY CLEAR 1, the number 192Ø will be replaced by Ø.

CLEARING ALL MEMORY REGISTERS

If you want to clear all ten memory registers at once, type

M C A

The calculator, protecting your interests, asks you to confirm that you really want to erase all ten memory registers, and then, if you type Y for yes, proceeds to clear all memories to Ø.

Remember that you can cancel memory operations before they are completed by pressing the space bar (or, when asked, as in this case, by typing N for no). When clearing all memory registers, be very careful that none of the registers contain anything you may need later.

MEMORY ARITHMETIC

Let's say you wish to figure your budget, and that you have the following budget categories:

Food
Clothing
Shelter

You are going through your check stubs and want to keep a running tally of each expense category and a running tally of all expenses.

Here are the expense items from your checkbook:

ITEM	CATEGORY	AMOUNT
groceries	food	$30.24
dry cleaning	clothing	$7.13
lunch	food	$4.13
house payment	shelter	$286.40
dinner	food	$27.86
laundry	clothing	$6.50
property tax	shelter	$1435.00
school clothes	clothing	$126.36

Let's assign memory registers as follows: 0 is for food, 1 is for clothing, and 2 is for shelter. It is far easier to keep track of what memory is holding which category if the name of the category is displayed. (In a following section, we will discuss the labeling of memories in detail.) For now, type

```
M V L
```

If the labels displayed are not FOOD, CLOTHING, SHELTER, then proceed with the following: type

```
M C L
```

To clear the current labels. If asked to confirm, type Y for yes. Now type

```
M T A
```

to type all the category labels. Type in the title, EXPENSES, and press the RETURN key, and the labels, FOOD, CLOTHING, SHELTER, pressing the RETURN key after each. Then press the space bar to cancel further label typing.

Having the proper labels displayed on the screen, let's clear the memories by typing

```
M C A
```

and the stack with:

```
C S
```

Clearing the stack is not necessary, but it lets us begin with a clean slate.

The X register will hold the running totals. The first step is to type in item 1, the groceries:

```
30.24
```

Do not ENTER^ the number. Instead store it in memory Ø by typing

M S Ø

As you can see by the cursor location, storing a number in memory has the effect of preparing the stack to lift when the next number is entered. Therefore, we needn't press ENTER^, but can move directly to the next item; type

7.13
M S 1

Now add the first two items together by pressing the + key. Then you can type the third item, lunch:

4.13

We don't want to Store the 4.13 in memory register Ø; doing so would destroy the 3Ø.24 that is there now. Instead, we want to add the 4.13 to the current contents of memory register Ø. To do that, type

M + Ø

In this case, M S + Ø will also work, but the S isn't absolutely necessary. By leaving it out you save one keystroke.

This leaves $34.37 in memory register Ø. Now we can add the X and Y registers. If you forget to add the cost of an item to its category before adding it to the grand total, you can recover the item from the LAST X register and finish the calculation by typing

L
M + <memory number>
CX

You can then continue. If you add a number to the wrong memory register, type

M - <wrong memory number>
M + <correct memory number>

If you fail to add an item to the grand total before introducing the next item, don't worry. Just store the latest item in the correct memory register and then press the + key twice: once for each number.

Work your way through this problem, continuing just as we have. When you are through, the registers should look like this:

```
Y              Ø
X          1923.62

LAST X      126.36

Ø            62.23
1           139.99
2          1721.4
```

When you have finished, you can easily cross-check the totals by typing

```
M R Ø
M R 1
+
M R 2
+
```

The answer should match the 1923.62 you obtained in your earlier calculations.

Besides adding to and subtracting from the contents of memory registers, you can multiply or divide the contents of memory registers by the contents of the X register. The commands are

```
M S * <memory number>
M S / <memory number>
```

ADVANCED MEMORY ARITHMETIC

In addition to carrying out arithmetic on a single memory register, you can affect all memories at once.

Let's say that the expense results now in the memories represent an average one month's expenses in each category. To find out what these expenses would look like for a whole year, just multiply them all by 12. Type

```
12 M * A
```

to return the memories to their original amounts, type

```
M / A
```

You can similarly add or subtract the contents of all memories simultaneously by typing

```
M + A
M - A
```

These four commands, as simple as they are to use, add a whole dimension (quite literally) to the power of the calculator. They are not found on other RPN calculators simply because you cannot easily see inside the memories on other RPN calculators.

MEMORY ARITHMETIC ERRORS

There are only two error conditions that can occur with memory arithmetic. The first is caused if you try to divide by zero, and the second is caused by overflow (too high or too low a number). Both errors will be discovered and displayed on the screen, and all registers will be left with their contents the same as they were before the error occurred.

DISKETTE MEMORIES

You can store up to ten memory sets, complete with titles and descriptive labels, on the APPLE HOW TO! diskette. Once there, they can be recalled or exchanged with other memory sets. Their labels can be independently recalled and stored. To view the list of memory sets currently stored on the diskette, type

M V D

If there are any memory sets currently stored, they will be named following each number in the left-hand column. After looking at the list, press the space bar to continue with the program.

STORING TO DISKETTE

To store the current memory set on diskette, type

M S D

If the memory set is untitled, you will be asked to title it. If there is already a memory set on diskette by that name, you will be asked if you wish to save the new set in its place. If there is no memory set by that name, or you do not wish to lose the old memory set, you will be shown the list of memory sets so you can select a new title to store it under. The questions asked of you when you save will depend on the conditions at the time; for example, you will not be asked if you wish to destroy the old memory named EXPENSES if, in fact, there is no old memory named EXPENSES.

As with all other memory operations, you can cancel the operation at any time by pressing the space bar.

RECALLING FROM DISKETTE

To recall a memory set from diskette, type

M R D

The program will ask any necessary questions and then recall the memory set from diskette. Data or labels currently stored in the Apple's internal memory that you wish to retain should be stored on diskette before recalling the diskette memory. (If there are data or labels in memory, the program will ask before deleting them.)

EXCHANGING WITH DISKETTE

You can exchange the internal memory set with any memory set on the diskette. This is a particularly useful feature if you want to keep working on the set you have, but need to look at a set on the diskette. By swapping the other set in, looking at it, storing any data needed in the stack, and then swapping your original set back in, you end up with your original set back in the Apple and the other set back in its place on the diskette. Then if, for example, you decided to use the other set's labels in your memory, you can recall just the labels, as will be explained below in RECALLING LABELS FROM DISKETTE.

To exchange the current memory with a diskette memory set, type

M X D

The program will ask any necessary questions and then exchange the two memory sets.

MEMORY LABELS

One of the chief weaknesses of pocket and desktop calculators with multiple memories is the unrealistic demand that they place on the user's memory. Can you really be expected to remember what categories of information you are storing in ten different memories? Most people end up using a large number of memories only when they absolutely must, and then they usually have to write down a list of the categories on a piece of paper.

The RPN Calculator, on the other hand, allows you to conveniently enter, change, or delete labels on any or all memory registers. And you can save up to ten sets of memories with labels on the diskette, so if you do regular budgeting or accounting, for example, you can simply recall a particular list of labels from the diskette.

VIEWING LABELS

The command to allow you to view the memory labels instead of the command lists is

M V L

for Memory View Labels. You can display the command lists again by pressing the W key. When you go back to the command lists again, the labels you have entered are not lost. Typing M V L again will switch you back to viewing the labels again.

TYPING A LIST OF LABELS

Let's say you want to do a problem that uses the categories

APPLES
ORANGES
PEARS
PLUMS

TOTAL

First type

M T A

for Memory Type All. You will find that the cursor has been placed at the beginning of the title line. To enter the title, type

FRUIT

and press the RETURN key. Let's say we want the fruit labels to be in memory registers Ø through 3 and the total to appear in memory register 5. To enter the first label, type

APPLES

and press the RETURN key. "APPLES" is now entered as the label for memory register Ø and the cursor has dropped down to memory register 1, waiting for the next label. Continue entering the ORANGES, PEARS, and PLUMS labels in this manner.

We now wish to include a line with no label. There are two ways to do that. One is to press the RETURN key. When you press the RETURN key without entering any characters, the Apple assumes you wish the label to remain as it is. This is fine in this particular instance because the label is now blank, just as we want it. If, however, the line already contained a label such as GOPHERS, you would probably want to remove it from your list of fruits.

To erase the current label, type

/

If you begin a label and change your mind, use the back arrow to back up to the beginning of the line, then type / and press the RETURN key. You do not need to use the RETURN key if you use the / key before typing any characters. Now go ahead and enter the last label, TOTAL, to finish off the list.

After you have entered the last label, you can cancel label entry by pressing the space bar at the beginning of the next line.

TYPING A SINGLE LABEL

Now let's put a new label, GRAND TOTAL, on memory register 6, and then remove ORANGES from the list. Type

M T A

Again the question mark will appear at the memory title. We want to keep the title the same, so simply press the RETURN key. We also want to retain APPLES, so press the RETURN key again. When the question mark is in front of memory register 1, now labeled ORANGES, type

/

and then press the RETURN key. (The instruction message reflects what the current label reads.)

Now step through the memory labels with the RETURN key until the question mark lies on the line for memory register 6. Then type the new label

GRAND TOTAL

Now press the RETURN key, and when the question mark lies at memory 7 press the space bar to exit from the label entry mode.

There is a way to add or change a single label without going through the entire list. Simply type

M T <memory number>

and then the label you wish to put there. For example, to change the label on memory 2 from PEARS to PEACHES, type

M T 2

and then type

PEACHES

This will replace the old label with the new one and immediately return you to the calculator mode.

To erase a single label, type

M T <memory number> /

(If this does not erase the label immediately, press RETURN.)

You can cancel an entry that is in progress by backing up to the beginning of the line and pressing the RETURN key to retain the old label.

RECALLING AND SAVING LABELS

You can recall labels from and save labels on diskette in much the same way as recalling or saving the entire memory. You cannot exchange the set of labels in memory that appear on the screen with a set of labels stored on the diskette. To recall a set of labels, you type

M R L

The program will display the diskette memory titles and ask you which set of labels you wish to recall. If you already have labels within the program, you will be asked if you wish to delete them. If you choose not to do so, the operation will be cancelled. You can then proceed to save the current labels in a different memory and then recall the set you wish.

Normally, labels are recalled from diskette for one of two reasons: either to edit them, or to use them for a new set of data. If you have recalled them to edit them, you can store them back on the diskette when finished by typing

M S L

You will be asked if you wish to delete the old label set, and, if you answer Y for yes, the new set will replace it.

In order to make it difficult to accidentally associate a new set of labels with the wrong memory set, the calculator requires that you title the new label set the same as the old memory set before you attempt to save the new label set.

If you have recalled the labels because you want to start a new set of data, be sure to change the title, using the command MTT, for Memory Type Title. This will prevent the destruction of the old memory set when you later save the new memory set on diskette.

OTHER MEMORY KEYSTROKES

The calculator is particularly forgiving about the order in which you type the letters in your memory commands. For example, to title the internal memory, you can type

```
M T T
```

or

```
M L T T
```

To view the labels, you can type

```
M V L
```

or

```
M L V
```

Whenever the meaning of the keystrokes you type in is unambiguous and not hazardous (typing A for all is not allowed before the action you wish to take), you can type the command letters in any order. If you get in trouble, it will be obvious from what the screen says, and you can just press the space bar to cancel.

OTHER SINGLE KEYSTROKE COMMANDS

There are a few RPN single keystroke commands that don't fit into the sections discussed so far. This section elaborates on those commands.

EXPONENTS

RPN makes exponents easy with the exponent key, ^ (the N key on your Apple II keyboard). (The exponents discussed here are not to be confused with the exponents of 1Ø used in scientific notation.) Let's work out the following problem:

```
(14 ^ 4) * 36
```

First solve 14 ^ 4,

```
14 ENTER^
4 ^
```

and then multiply the result by 36,

```
36 *
```

to give an answer of 1,382,976. (Keep in mind that the arrow [^]

after ENTER is just part of the Enter command name, short for "enter and raise up in the stack".)

FINDING ROOTS, SQUARE AND OTHERWISE

You can calculate the square root of the number in the X register at any time by pressing the Q key. The numbers in the other registers will not be affected by this operation. For example, to find the square root of 64, type

```
64 Q
```

If the number 64 was already in the X register as the result of a previous calculation, you would type only Q .

Finding roots that aren't square is almost as easy as finding square roots. To find the 16th root of the number 65536, type

```
65536 ENTER^
16 R
```

Try the 3rd root of the number 27:

```
27 ENTER^
3 R
```

PI

The remaining single keystroke command is Pi. Pressing the P key will place the value 3.14159265 in the X register. To calculate the circumference of a circle with a diameter of 12 inches, for example, you would type

```
12 P *
```

The answer, 37.6991118 would then appear in the X register. (37.70 if the display were fixed to 2 places.)

This completes the discussion of single keystroke commands. We will now continue our exploration of the RPN Calculator with the Function commands.

FUNCTION COMMANDS

Move to the COMMANDS: 3 list by pressing the W key. As you can see, all the function commands are preceded by an F. Another way to display the COMMANDS: 3 list, when not displaying memory labels, is to press the F key. The function menu will automatically be displayed until you have finished entering the function. Then the previously used COMMANDS: list will be displayed again. It takes some time to display the command lists. So if you are doing a lot of function

commands and have pretty well learned COMMANDS: 1, set the list at COMMANDS: 3 with the W key.

If you have the memory labels displayed and wish to see the COMMANDS: 3 momentarily, then type

```
F W
```

The list will be displayed, and, having already typed the F, you can type the rest of the desired command.

You can cancel any function operation, before it is completed, by pressing the space bar.

SETTING THE DISPLAY

You can format the display of the X register to round off to as many significant digits as you wish, up to a maximum of eight. When the calculator program is first run, the X register is set to round to two decimal places. To change the number of decimal places, type

```
F D <number of digits>
```

The accuracy of the calculation will not be affected. The computer will still be using the full nine-digit precision in making calculations. Only the number of digits you see in the X register is affected.

PERCENTAGES

Typing

```
F %
```

will calculate X percent of Y. For example, if you wish to find out what 28 percent of $1436.ØØ is, type

```
1436 ENTER^
28 F%
```

The answer, $4Ø2.ØØ, will be revealed.

It is unnecessary to press the SHIFT key when typing %. If you do use the SHIFT key, however, the command will still work.

THE INTEGER FUNCTION

To recover the integer portion of a number use

```
FI
```

For example, to find the integer part of 138.38, type

```
138.38 ENTER^
FI
```

THE FRACTION FUNCTION

The fractional portion of a number can be recovered by typing

```
FF
```

For example, to find the fractional part of 138.38, type

```
138.38 ENTER^
FF
```

The answer, .38, will be displayed in the X register.

TRIGONOMETRIC FUNCTIONS

The trigonometric functions used by the RPN Calculator assume that the angles are given in radians. The six trigonometric functions provided by the calculator are

```
FS -  Sine
FAS-  Arc Sine (inverse sine)
FC -  Cosine
FAC-  Arc Cosine (inverse cosine)
FT -  Tangent
FAT-  Arc Tangent (inverse tangent)
```

These trigonometric functions act only upon the contents of the X register.

LOGARITHMIC FUNCTIONS

The RPN Calculator has 4 logarithmic functions:

FL - the common or base 1Ø log of the value in the X register.

FX - the common or base 1Ø antilog of the value in the X register. It raises 1Ø to the power of the value in the X register.

FN - the natural log of the value in the X register. Rather than base 1Ø, it works in base e, or 2.71828183....

FE - the natural antilog of the value in the X register. It raises E to the power of the value in the X register.

RECIPROCALS

FR, the reciprocal function, divides 1 by the value in the X register. It does not affect the contents of any other register.

To find the reciprocal of 4, for example, type

```
4 FR
```

The answer, .25, now appears in the X register, and the 4 appears in the LAST X register.

FACTORIALS

An integer factorial is the product of all integers from one through that integer and is symbolized as an integer followed by an exclamation point. For example, the factorial of 5, 1*2*3*4*5, is written

```
5!
```

To get the answer, type

```
5 F!
```

Factorials can be figured for any positive integer from Ø to 33. If 33 seems like an awfully low number, try doing some factorials, starting at 5 and going up.

PRINTING

Please select COMMANDS: 4 by pressing W until it appears. If your Apple is equipped with a standard type of printer and an Apple intelligent interface card, the RPN Calculator will print results for you. In order to do so, however, the program needs two pieces of information from you: the slot into which your printer is plugged, and whether your printer needs line feeds in addition to carriage returns.

To set the printer slot number, press CTRL-N for new slot number. (To type a control character, hold down the CTRL key, then press the other key, in this case, N. After releasing the N key, release the CTRL key.) Now type the number of the slot. If this number is different from the number shown on the command list, the diskette will be updated with the new slot number.

Some printers will not advance up a line at the end of each line printed unless they receive a special character called a line feed. You probably know what your printer's requirements are. If not, just try printing with the line feed set on "LF NOW OFF". If all the printing overlaps on a single line, then set the option to "LF NOW ON" by pressing CTRL-F. As with the slot number, this information will be stored on the diskette.

If a printer is not attached to your Apple, don't try to use RPN'S print commands. If you do, the Apple may quit responding to the keyboard, and you may have to reboot the APPLE HOW TO! diskette to recover. You can avoid having this happen by setting the slot number to Ø.

RPN PRINTING COMMANDS

The Apple RPN Calculator has 3 different printing modes: Manual, Normal, and All. The current printing mode is displayed on the top line of the screen at all times. It should now say

PRINTER: MANUAL

When the RPN Calculator is in this mode nothing will be printed unless you press one of the manual print keys explained later. You can change the printing mode by pressing CTRL-P. Do that now. The line at the top of your screen now says

PRINTER: NORMAL

This mode prints each step of the calculation sequence, printing each time a function or calculation occurs. The actual results of the function or calculation will not be printed (except on the screen) unless you press CTRL-X to print the contents of the X register.

Press CTRL-P again, and

PRINTER: ALL

will appear at the top of your screen. In this mode, numbers you enter, functions, and intermediate and final answers are printed as they appear on the screen. Three asterisks to the right of a printed number indicate that the number is the result of an operation.

To do a quick test of your printer, command the calculator to print out the contents of all registers by pressing

CTRL-A

(holding down the CTRL key while pressing the A key). If the printer responds, you can proceed. Otherwise, recheck the printer and confirm it is plugged into the slot indicated under COMMANDS: 4, is plugged in to the AC outlet, and is turned on.

The following print commands are available with the RPN Calculator:

CTRL-X	prints the contents of the X register
CTRL-S	prints the contents of the Stack: the X, Y, Z, and T registers
CTRL-R	prints the labels and contents of all memory Registers
CTRL-A	prints the labels and contents of All registers, both stack and memory
CTRL-J	advances the printer paper 1 line
CTRL-L	(with some printers) advances to end of page
CTRL-T	Types notes directly (see next section)

Try filling the calculator with numbers and labels and then printing them out using the various commands.

If, when you try to print something, the calculator does not print and quits responding to the keyboard when you tell it to print, check that your printer is turned on and is plugged into the correct slot. This is often the problem.

PRINTING NOTES DIRECTLY

How many times have you stared at some numbers on a piece of adding machine or calculator tape, wondering what they represent? With the RPN Calculator, you can print directly on the paper, putting down whatever impromptu notes you feel are important. To do so, first press

CTRL-T

for Type. The computer will acknowledge the instruction by telling you to go ahead and TYPE NOTE TO BE PRINTED. The bottom line of the screen will be cleared, and the cursor will be positioned in the bottom left-hand corner. Type the note you wish to print, and then press the RETURN key. The printer will respond by printing your note. The note can be up to six lines long, but all but the bottom line will be invisible on the screen. Normally, notes are going to be a matter of 1Ø or 2Ø characters, but if you reach the end of the screen and the first 4Ø characters scroll off, don't panic; they are still there. Just finish your note and press RETURN.

If you change your mind after pressing CTRL-T, press RETURN before entering any characters and the operation will be cancelled.

With a little practice, you will soon be adept at using Apple's RPN Calculator. Besides its uses as a teaching aid, you will find that the RPN Calculator program is a very useful everyday tool, in many ways more useful than a standard calculator.

SUMMARY OF RPN COMMANDS

COMMANDS: 1

Command	Function	Page reference
Return	enter and push stack up	6
CX	Clear X	6
CS	Clear Stack	9
L	Last X	12 - 13
P	Pi	29
+	add	6 - 8
-	subtract	6 - 8
*	multiply	6 - 8
/	divide	6 - 8
E	enter Exponent	14
S	change Sign	6
R	Xth Root Y	29
Q	sQuare root X	29
U	roll Up	10
D	roll Down	10
X	eXchange X and Y	10
A	toggle Audio	10
K	toggle Key clicks	10
ESC ESC	end	4
W	move to next command list	4

COMMANDS: 2

Memory Commands

0-9	used in certain memory commands to refer to the number of the individual memory you wish to affect	16
A	All (used in certain memory commands to refer to all memories)	16
D	Diskette (used in certain memory commands to refer to list of memory sets stored on diskette)	16
L	Label (used in certain memory commands to refer to memory labels)	16

Command	Function	Page reference
MC?	Clear Memory ?	19
MS?	Store Memory ?	17
MR?	Recall Memory ?	17 - 18
MX?	eXchange X-register contents with Memory ?	18 - 19
M +,-,*,/	Memory arithmetic	19 - 23
MSL	Store Labels on diskette	27
MRL	Recall Labels from diskette	27
MT?	Type label ?	26 - 27
MTT	Type Memory Title	27
MVL	View Labels on internal Memory set	25
MVD	View titles of Memory sets stored on Diskette	23
space bar	cancel command	16

COMMANDS: 3

Function Commands

FD	Fix Display (used to control the number of decimal places displayed in the X register)	13, 30
F%	calculate X percent of Y	30
FI	recover the Integer portion of a number	30 - 31
FF	recover the Fractional portion of a number	31
FS	Sine	31
FAS	Arc Sine	31
FC	Cosine	31
FAC	Arc Cosine	31
FT	Tangent	31
FAT	Arc Tangent	31
FL	Log	31
FX	common antilog of X register	31
FN	Natural log	31
FE	natural antilog of X register	31
FR	Reciprocal	31 - 32
F!	Factorial	32

COMMANDS: 4

Print Commands

Command	Function	Page reference
CTRL-P	change general Print mode	33
CTRL-X	print contents of X register	34
CTRL-S	print contents of Stack	34
CTRL-R	print conetents of memory Registers	34
CTRL-A	print All registers, both stack and memory	33
CTRL-T	Type notes	34
CTRL-J	advance paper	34
CTRL-L	page feed	34
CTRL-N	New slot number	32
CTRL-F	toggle auto lineFeed	32

CHAPTER 3
ROD'S COLOR PATTERN

Programming Language: Integer BASIC

Rod's Color Pattern program puts colorful kaleidoscopic designs on your video screen. However, this program isn't just another pretty picture generator. By modifying various statements in the program and then running the revised program, you can learn a great deal about programming in Integer BASIC.

If you are new to BASIC programming you should read the Apple II BASIC Programmming Manual if you have not already done so. This manual teaches you the fundamentals of Integer BASIC programming that you need to understand Rod's Color Pattern program.

USING ROD'S COLOR PATTERN

After you have selected Rod's Color Pattern from the diskette menu by choosing option number 2, the program menu appears on your screen. The program menu gives you 3 options:

```
1 WATCH THE PATTERN
2 LIST THE PROGRAM
3 END
```

Before you start looking at the program listing, you should choose option 1 to see what the program actually does. When you choose this option the program menu disappears, and the screen begins to fill with color -- or shades of black, white and grey if you have a black and white monitor. The patterns on the screen will keep changing until you press the space bar to return to the program menu.

Once back at the program menu you can get down to the business of learning more about BASIC programming by choosing option 2. When you choose option 2, the program menu again disappears, but this time the screen fills with a BASIC program listing.

The listing on your screen is the first few lines of Rod's Color Pattern program. Notice that the program listing consists mostly of REMark statements. You will find that this holds true throughout the program listing. These REM statements describe in detail what each of the program lines does. The REM statements enable you to understand how Rod's Color Pattern works and how you can change the program to create different patterns.

Near the bottom of the screen the instructions

PRESS SPACE BAR TO CONTINUE

and

PRESS THE "ESC" KEY TO RETURN TO MENU

are displayed in inverse video (black on white). If you want to stop the program, simply press the ESC key as indicated; the program menu will return to the screen. If you press the space bar, the next few lines of the program will be displayed. Each additional time you press the space bar, the next sequence of program lines will appear until the entire program listing has been displayed. When you reach the last frame, you will be instructed to

PRESS SPACE BAR TO RETURN TO MENU.

CHANGING THE PROGRAM

Rod's Color Pattern is not a protected program. You can move it to another DOS 3.3 diskette so that you can work with the program and make changes. To do this, first boot the system with the diskette to which you want to move the program. Then, remove the other diskette and put in APPLE HOW TO! Type:

LOAD ROD'S COLOR PATTERN

When the disk drive stops whirring and the red light goes out, remove APPLE HOW TO! Then insert the diskette on which you want to save the program and type:

SAVE ROD'S COLOR PATTERN

With the program transferred to the other diskette, you may proceed to experiment with the program and save the results.

If you want to create your own color pattern program, you can use Rod's Color Pattern as a model. If you have a printer connected to the Apple, you can print on paper a listing of the program you have transferred to your new diskette (see above). This paper copy of the program will prove very useful because you can refer to it while your screen is displaying your own programming efforts.

To print a listing of Rod's Color Pattern program, you must first exit from the program by choosing the END option from the program menu. You will be left in Integer BASIC as indicated by the angle bracket prompt.

From Integer BASIC type

```
LOAD ROD'S COLOR PATTERN
```

Then initialize your printer. For most printers you can simply type

```
PR#
```

followed by the slot number into which your printer interface card is plugged. (If you aren't sure how to initialize your printer, consult your printer manual.) Once your printer is initialized, type

```
LIST
```

Your printer should now begin to print the listing you need. If it doesn't work, check to be sure that your printer is installed correctly and that it is turned on.

With your program on your own diskette and a paper copy before you, you are prepared to experiment to your heart's content.

CHAPTER 4
SCROLLING WINDOW TUTOR

Programming Language: Integer BASIC

The Scrolling Window Tutor consists of six easy lessons that instruct you in the use of text windows. "What's a text window?" you might well ask. The program will answer that question for you and give you the information you need to use text windows in your BASIC programs.

To use this program successfully, you should have some knowledge of BASIC programming.

Choose SCROLLING WINDOW TUTOR from the diskette menu. The disk drive will whir and click for a while, and then a "welcome screen" will appear. The welcome screen will ask you whether you would like to use the Scrolling Window Tutor's built-in sound effects. Answer Y for yes, or N for no, and press RETURN. Then the "title page" will introduce the program, (the title page is a good example of how text windows can be effectively used) and a list of options, called a "program menu," will appear.

If you haven't used Apple's text windows before, you should choose

1. RUN TUTORIAL FROM THE BEGINNING

from the list of options. The program will then lead you step by step through the window-making process, quizzing you along the way to make sure you understand the concepts presented. You can quit the program and jump to the review point menu at almost any time by pressing the ESC key. (This will not work in some places when you are being asked a yes/no question.)

When you have completed all the lessons, the review menu will appear on your screen. This menu is the same as the menu displayed when you choose

2. GO DIRECTLY TO REVIEW POINT

from the main program menu. The options listed let you review any lesson in the Scrolling Window Tutor program. Option 9, RERUN ENTIRE PROGRAM, gives you the opportunity to change your mind about whether you want the sound effects to be turned on.

CHAPTER 5
ASSEMBLE-IT-YOURSELF MULTI-TONE KIT

Program Language: Integer BASIC

If you're like most people who've dabbled with computers, you've heard the terms "assembly language" and "machine language" but know virtually nothing about them except that they contain a lot of strange-looking commands that, at first glance at least, seem totally nonsensical. The aim of the Multi-Tone Kit program is to teach you a bit about writing programs in these languages and to give you some appreciation for their usefulness. But that's not all. By progressing through the steps in the Multi-Tone Kit, you also will be privy to a little-publicized method of using your Apple's game paddles to create some truly exotic sounds sure to impress even the most well-travelled ear.

To understand every aspect of the Multi-Tone Kit, you should have a bit of experience using your Apple and writing programs in BASIC. Still, even if you're brand new to computers, you shouldn't hesitate to run the Tone-Kit program. Just don't expect everything to be crystal clear the first time through.

Before embarking on our journey, a few definitions are in order.

> Machine language is the computer language that controls the lowest level or fundamental internal operations of the computer.
>
> The Monitor is a program that allows the user to initiate machine language instructions.
>
> Assembly language is a computer language made up of simple words and abbreviations that can be easily remembered yet quickly and easily converted to machine language by the computer's assembler.
>
> The Assembler is a program that converts assembly-language instructions into machine-language instructions.

These terms, as well as some other terms used in the Multi-Tone Kit, are included in the glossary at the end of this chapter.

BEFORE YOU BEGIN

Choose

4 ASSEMBLE-IT-YOURSELF MULTI-TONE KIT

from the main diskette menu. The program will announce itself and present you with the program menu.

Just below the program title and the program author's name, you will find a couple of instructions. For now, you can ignore the instruction about PDL(Ø) and PDL(1). The next instruction tells you to press the ESC key to redisplay the menu. This information may come in handy in an emergency.

Below this, you will find the program menu. It consists of the following options:

1) CONQUER FEAR OF MONITORS (TUTORIAL)

2) INVOKE MULTI-TONE GENIE (MAGICAL)

3) END (TERMINAL)

A word of caution: do not select number 2 before selecting number 1. In number 1, you will travel to a program built into the Apple called the Monitor that will allow you to construct the Multi-Tone Kit. Afterward, you can command the Multi-Tone Genie to reconstruct the kit for you. But, invoking the Genie before passing through the Monitor will not only cause crabgrass to grow in your front lawn but will also eliminate much of the understanding you can gain by using your Apple to construct your own personal music-maker. Monitor passage, it must be remembered, is a test of personhood and should not be taken lightly.

Playing the Multi-Tone instrument is a job for a musician, but assembling it is a job for a programmer. "Oh, no!" you say, "I've never done machine-language programming in my life! Why, I can't even be trusted with BASIC!" Not to fear. You will be in complete control of command-central at all times. If at any time panic overtakes you, pressing the ESC key will instantly return you to the main menu.

No experience is expected -- or even wanted, for that matter. This is a journey for anyone with a sense of adventure, a lust for the hunt.

AND WE'RE OFF...

Please select

1 CONQUER FEAR OF MONITORS

from the menu, and then press the RETURN key. A window outlined in asterisks now occupies the top half of the screen. Information and instructions will scroll through this window. The second line of text from the bottom of the window displays commands for you to carry out. The window's bottom line carries this permanent reminder:

PRESS ESC TO RETURN TO MENU

The actual machine language assembly will occur in the bottom half of the screen where you now see the Integer BASIC prompt, > , and the blinking cursor.

Even though it appears that the program has ended, as evidenced by the > prompt and cursor, it has just begun. Follow the command and

WATCH BELOW AND PRESS SPACE BAR

After reading the contents of this frame, follow the next command and press the RETURN key to drop into the Monitor. When the asterisk prompt appears at the bottom of the screen, you know that you are in the Monitor.

IF YOU HAVE THE AUTOSTART ROM

If your Apple is equipped with the autostart ROM, you normally end up in BASIC when you press the RESET key. The next step in building the Multi-Tone Kit is to set the programming environment so that pressing the RESET key will leave you in the Monitor, just the way the old Monitor ROM does. This feature is invaluable when you are programming in machine language since it prevents your having to keep typing CALL - 155 every time you press the RESET key (which you do a lot down here).

The Monitor instructions to make RESET leave you in the Monitor are:

3F2:69 FF 5A

For now, the Multi-Tone Genie will type in the instructions for you, but you may want to type in these instructions yourself some other time that you are working in the Monitor. This change is purely temporary; when you are finished with the Monitor and boot up again, RESET will once again leave you in BASIC.

If you do not have an Autostart Monitor ROM, RESET will automatically leave you in the Monitor, and the Genie will not ask you to set the environment.

FRAME 1

You will notice that the number 1 has appeared in the upper left-hand corner of the box. This number refers to the expanded explanations which appear below. Each time you see a new number appear, you can read more about what is going on beside the corresponding number in this manual. You can easily get through the program without using the manual, but if you do not have a lot of experience in programming, using 65Ø2 machine language, or using the Apple computer, you will glean a lot of useful and, hopefully, interesting information from it.

Hidden inside Apple's Integer BASIC is a tiny language called the Mini-Assembler. This program was used during the development of the BASIC language itself, and, while it is very limited compared to the assembler found in the DOS TOOL KIT from Apple, it is a useful way to catch a glimpse of the world of super-high-speed programs.

Computer memories are much like post office boxes. Each one of the possible 65,535 memory cells, or bytes, in an Apple has its own private address number. If you type the address of the very first byte of a program and then type the command "G" for go, that program will begin running. An example that is familiar to many disk users is a special program that starts at address 3DØ. This program will, from the Monitor, reconnect DOS and reinstate BASIC. Typing 3DØG tells the Apple to Go to 3DØ and begin whatever program is to be found there.

In the case now before us, we will tell the Apple to go to the post office box marked F666 and begin whatever program it finds there. In this instance it finds the Apple Mini-Assembler.

FRAME 2

The starting location or address of the program you will write with the help of the Multi-Tone Kit, CØØ, was selected because it is pretty well out of the way, so Integer BASIC is unlikely to "over-write" and thereby erase it. There are other locations that would have done as well.

The $ printed before the CØØ in the scrolling text window indicates that CØØ is a hexadecimal number. Hexadecimal numbers are discussed in detail in the next frame.

FRAME 3

This discussion introduces you to the hexadecimal number system. Hexadecimal is a fancy word for a number system with 16 digit symbols instead of the 1Ø digit symbols we civilized folk use. The exact reason for computer people selecting the number 16 is a little beyond the scope of this manual, and in fact the choice of 16 is somewhat arbitrary. (Many of you may have heard the rumor that machine language programmers have 16 fingers. This is untrue. However, it would help if machine language programmers did have 16 fingers.) It is useful to note, however, that 16 is an even power of 2 (2 X 2 X 2 X 2) while 1Ø is not. Computers are wild about 2, so hexadecimal it is!

There are two aspects of hexadecimal that are worth talking about. The first is that because we aren't used to it, it is automatically scary. This can be overcome with time and patience. The second is that we only have 10 number symbols lying around: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. So if we want 10 through 15 to have single symbols all their own, we have to come up with those symbols. 'Way back in the history of computers, around the early 1950's, people struggled valiantly with this problem, inventing all kinds of strange squiggles and scrawls which to them appeared to embody the true personality of numbers such as eleven or twelve. These weird chicken-tracks did nothing but confuse people, so it was decided to borrow symbols we already know. Here is the complete set:

DECIMAL	HEXADECIMAL	
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
1Ø	A	(Here's the new stuff)
11	B	
12	C	
13	D	
14	E	
15	F	

You have now learned half of all there is to know about hexadecimal numbers. The other half has to do with "positional representation".

Now come out from under the bed! You've been using positional representation in numbers since the lst grade! It's just that no one ever used such a confusing term for it before. And they are unlikely to again, for that matter.

Now you've probably heard everybody bandying about the fact that zero is the most important number. Folks say the medieval Arabs and the Mayans were brilliant mathematicians because they each invented it. So what is so important about zero? A zero is nothing, and what does nothing have to do with positional representation?

Below is a number. What number is it?

1

If you guessed 1 you are both right and wrong. The digit is one, but the number is 1Ø . You see, right after the 1 is some nothing. Perhaps it would be clearer if we put the nothing after it so you could see it better:

1Ø

What positional representation says is that this 1 means ten because it is in the second column, the 1Ø's column. (The zero's job here is to let you know that the 1 is in the 1Ø's column.) If the 1 were in the third column, as in the number 162, the 1 would represent one hundred. (Not all number systems are like this. For example, in Roman numerals, the symbol "L" means 5Ø — it doesn't matter where you write it, it means 5Ø. In MCMLXXXX it means 5Ø; in CL it means 5Ø. Its position is unrelated to its absolute value.)

The only difference between hexadecimal numbers and our familiar decimal numbers is in the "value" of each column. In both cases the first column (the one on the right) is the 1's or units column. You multiply the digit found there by 1 to get its value. Digits in the second column of a hexadecimal number are multiplied by 16, just as they would be multiplied by 1Ø in the decimal system. The third column in a hexadecimal number is multiplied by 256 (the square of 16, just as 1ØØ is the square of 1Ø), and the fourth column is multiplied by 4Ø96 (the cube of 16).

All this new information may seem a little confusing. Let's look at a concrete example, the familiar 3DØ. Moving from right to left:

The first column equals	Ø times	1 =	Ø
The second column equals	D (or 13) times	16 =	2Ø8
The third column equals	3 times	256 =	768
Adding the column values gives the result			976

Another way to express this same concept is like this:

In base 1Ø

$976 = (9 \times 1Ø^2) + (7 \times 1Ø^1) + (6 \times 1Ø^0)$

In base 16

$3DØ = (3 \times 16^2) + (D \times 16^1) + (Ø \times 16^0)$

Therefore, giving the BASIC command, CALL 976, is equivalent to giving the Monitor command, 3DØG. Both commands are simply calling upon the same numerical address (one with decimal notation, and one with hexadecimal notation) and telling the computer to execute what it finds there.

Our starting address of CØØ is easy to translate: there are no units and no 16's. C is 12, and it is in the 256's column; 12 times 256 = 3Ø72. After we have finished writing this program, you will find that you can start it just as well by typing either

```
CALL 3Ø72
```

or

```
CØØG
```

and now you know why.

The point of this discussion is to acquaint you with this new number system. You are not expected to suddenly be able to think in hexadecimal and begin doing mathematical computations in base 16. It is useful when you see a number like $F666 to realize that you are looking at a real live number, a number that is exactly 1 higher than $F665 and 3 lower than $F669. It will provide you with another piece of the computer puzzle and may make other things you have learned along the way suddenly make more sense.

FRAME 4

The 65Ø2 microprocessor is the heart of the Apple II and Apple II Plus computers. All computations and decisions take place within it. Everything else in the computer is there to provide the microprocessor with power, memory storage, and ways for it to talk to the outside world.

The memory storage in the computer is set up like an enormous room full of post office boxes. Each box can hold a single number (byte) at any one time, and any box can be instantly found by addressing its individual box number. Because of your ability to get at any box quickly and directly, this kind of memory is called Random Access Memory, or RAM. Your Apple has between 32 thousand and 65 thousand individual post office boxes in it. More on these boxes later.

Registers are places inside the microprocessor used for temporary storage of data and address information. There are three data registers inside the 65Ø2: the X and Y registers and the Accumulator. Each register can hold one number from Ø to 255. In addition, the number stored in the Accumulator can be added, subtracted, multiplied, or divided by a number stored either in RAM or in the X or Y register. The result of this calculation is then stored in the Accumulator, replacing its original contents. The microprocessor can also make

decisions based on what is held in the registers, deciding, for example, to do one thing if the result of a subtraction is zero and quite another if the result is not zero.

From thousands of such simple arithmetical and logical operations grow the powerful languages and programs that can turn the Apple into a word processor, a tax analyst, or a real-time animator.

Instructions are given to the microprocessor in the form of numbers. To us, these numbers look pretty much like any other numbers, but to the computer they represent specific commands. The job of an assembler is to make it easier for people to talk to the microprocessor by making these instructions a little more human. While the microprocessor only understands numbers, the assembler understands words. These command words are standard on any 65Ø2 assembler, so you won't be learning words now and having to forget them in favor of a new set later.

Words that are understood by the assembler are called mnemonics, a fancy term derived from a Greek word meaning "memory aid." The 65Ø2 command to jump to a different part of a program is the number $4C. The mnemonic for the jump command is JMP . Most people find JMP easier to recall than $4C .

In the program we are now writing, we are using the X register to hold the number of the paddle we are going to read. Since we want to read paddle Ø, we want to load the number Ø into the X register. We do so by typing LDX for LoaD X. After we've given the assembler the LDX command, we have to tell it what to load the X register with. We can either load it with an actual number, in this case, Ø, or we can give it the address of a post office box which has that number in it. The advantage to selecting a post office box, say box number 12, to hold the zero is that we can come along at any future time and change the number held in box number 12 from Ø to any other number and the program will instantly start using the new paddle number. However, because we always want to use paddle Ø in the Tone-Kit, we are putting the actual number Ø directly into a register instead of storing it somewhere else.

The fancy name for an actual number is an "immediate" number: the microprocessor can use the number immediately; it doesn't have to open a post office box to read what is inside. To tell the assembler that we want the immediate number Ø instead of the contents of post office box Ø (which could contain almost any number right now), we type a pound sign (#) before the Ø. The rule, then, is that to refer to an actual number, we type the pound sign (#); to refer to a particular address or post office box, we omit the pound sign.

FRAME 5

Take a look at what the assembler has done with our command:

```
ØCØØ-   A2 ØØ      LDX   #$ØØ
```

First, it has assigned to the command the address we asked for: CØØ. In the middle it has printed: A2 ØØ. The ØØ is simple; that is our paddle number. The A2 is the hexadecimal code which tells the microprocessor: "the number that follows is an immediate number that should be put directly into the X register." Last, it has printed the mnemonic and the immediate number: LDX #$ØØ. (The Mini-Assembler assumes all numbers are in hexadecimal notation. It will add a dollar sign whether you type it in or not. More powerful assemblers, such as EDASM, found on the Apple DOS Toolkit Diskette, allow you to enter either base 1Ø or hexadecimal numbers and will do any necessary conversions for you. Sigh.)

You can find all the codes and all the mnemonics in the Apple II Reference Manual supplied with your computer. If you look under LDX in the tables, you will see there are several different numbers, depending on whether the number that follows is an immediate number, an absolute address, or is supposed to be handled in some special way (zero page, Y; absolute, Y; etc.). When you use an assembler, you don't have to pick the number; the assembler figures out what you want and picks the number for you. All you ever have to remember is to LoaD X.

Although there are only a few pages of instructions for the 65Ø2, and each one does but a small operation, machine language quickly becomes complex and is not grasped with a simple glance. There are great rewards in terms of speed and pristine elegance in the use of machine language, but it is hardly necessary for most people to learn it. This program is intended to be fun; if these added explanations seem overly complicated, please skip them; you can finish your Multi-Tone Kit and it will function flawlessly without them.

The content of the window now displayed on the screen informs us that we can use a space "instead of typing in $CØ2". You might wonder what happened to $CØ1. It is being used to store our paddle number, $ØØ, in the first command. ($CØØ holds $A2 and $CØ1 holds $ØØ.) Each assembled line will start out with the announcement of the first address in the line, as in ØCØØ above. But 65Ø2 commands may use 1, 2, or 3 bytes depending upon whether they use numbers and how large (1 or 2 bytes) those number are. So you don't know where the next command should begin without adding 1, 2, or 3 to the previous location (in hexadecimal, of course). The assembler will be happy to figure it out for you, if you just type a space to let it know that you want it to.

FRAME 6

For the BASIC programmers in the group, a subroutine in machine language is very much like a subroutine in BASIC. To go to a subroutine in machine language, you use the mnemonic JSR for Jump to SubRoutine, just as you use the command GOSUB in BASIC. The JSR is followed by the address of the subroutine, in the same way that BASIC's GOSUB is followed by its subroutine's "address" — a line number. The difference is that in machine language you give the microprocessor the address of the actual post office box of the first byte in that subroutine, so it doesn't have to look it up. On the average, this lets the Apple find an assembly language subroutine more than 1ØØ times faster than a BASIC subroutine.

The command found at the end of a BASIC subroutine is RETURN; the command found at the end of a machine language subroutine is RTS for ReTurn from Subroutine. They are functionally the same.

FRAME 7

When you ask the Apple to read a paddle, it first sends out a charge of electricity to all four paddle circuits at once. (You can plug four paddles into an Apple, a little-known fact disclosed in your Apple II Reference Manual.) After this initial charge has been sent, the Apple goes and looks at the one paddle you've asked about and sees if the charge is still there. If it is, it counts 1 and goes and looks again. If the charge is still there, it counts 2. It keeps checking and counting until it finds that the charge has dropped or the count has reached 255, whichever occurs first. How long the charge takes to "bleed off" depends on where you have the paddle set: the more resistance there is, the longer it takes.

The Multi-Tone program takes advantage of the delay caused by this counting process: The higher the resistance of the paddle, the longer it will take the subroutine to finish counting, the slower the program will run, and, thus, the lower will be the frequency or the pitch of the sound produced.

Here is a general programming hint on the Apple: Because all four paddle inputs are "charged" when any one paddle is read, reading a second paddle immediately after a paddle is read will result in an incorrect reading. Therefore, it is prudent to use a short timing loop, as shown in this BASIC example:

```
1Ø X = PDL (Ø)
2Ø FOR I = 1 TO 1Ø: NEXT I
3Ø Y = PDL (1)
```

Of course, you don't have to have a simple timing loop; you can do a computation or graphics plotting or anything else during this period--anything that takes time.

There is no such delay loop used in the Multi-Tone Kit because we are not concerned with measuring the results of the reading; we are creating a variable time pause based on how long it takes to read the paddle. It is simply a trick.

FRAME 8

APPLE SWITCHES

There are a number of switches inside your Apple. To turn on one of the switches, you trip one button. To turn off that switch, you trip another button. Well, actually, the microprocessor trips the buttons.

Imagine that you have one of these two-button switches for turning on and off your color TV. But instead of having the two buttons mounted on your wall, you rent out two post office boxes with wastebaskets underneath and put one of the buttons in each of the boxes. ("Good grief, Martha, he's started up with the post office boxes again.") You connect each of the buttons to the door of its box, so each time the box is opened, the button is tripped.

Now, each time you want to turn on your TV, you drop a post card in the mail to that box number. The mailman opens the box to put the card inside, tripping the button and turning on the TV. The postcard drops in the wastebasket. Later, you turn off the TV by sending a package to the other box number. The mailman obligingly opens the door to that box, tripping the other button and turning off the TV. The package likewise drops into the wastebasket. You can also drop by in person to pick up your mail. Of course, there won't be any (it went in the wastebasket!), but, depending on which box you open, you switch your TV on or off.

Consider the BASIC and machine language commands for turning on and off the Apple's color graphics:

	In BASIC	In Machine Language
ON:	POKE -163Ø4,Ø	$CØ5Ø
OFF:	POKE -163Ø3,Ø	$CØ51

Memory locations $CØ5Ø and $CØ51 (-163Ø4 and -163Ø3 in decimal) are addresses like any other addresses in the Apple's memory. But these addresses are connected to special switches called soft switches. Any time the 65Ø2 attempts to Load or Store the contents of one of its registers from or to these particular memory locations, the switch hidden behind them is tripped. As in the analogy above, it

doesn't matter what you attempt to Load from or Store in the box -- it can be Ø, 1Ø, or 255. All that matters is that the address has been accessed.

When you have finished with the Multi-Tone Kit and have booted up again with an unprotected diskette, try typing the color graphics addresses while you are in the Monitor. You will see the switches go on and off as your screen flashes first color, then text.

A list of all the soft switches for graphics and sound control is included in the APPLE II Reference Manual that was supplied with your computer.

The switch we are using to toggle the speaker takes up only one memory location instead of two. Each time the address is accessed, the speaker cone moves alternately up and down. If the speaker cone moves often, the sound produced has a high frequency or pitch; if there are long delays between movements, the sound has a low frequency or pitch. If the cone is moved up and then down only once, it makes a single click.

MORE NUMBERS

Some of you were no doubt somewhat baffled when you figured out that

$CØ3Ø = (C [12] * 4Ø96) + (Ø * 256) + (3 * 16) + (Ø * 1)

or 492ØØ, not -16336, as the Multi-Tone Kit program states. An explanation follows. The explanation is not being billed as simple or reasonable because it isn't. It is just the way it is.

Many computer languages, Apple's Integer BASIC among them, do not allow use of numbers higher than 32767. Integers are stored in two adjacent bytes or memory cells. A double byte of memory can store numbers as small as $Ø and as large as $FFFF, which works out to 65535 in decimal. The problem is that if you interpret the numbers this way, there is no way to have negative numbers. So the contents of two bytes are interpreted like this:

Ø, 1, 2, 3, ..., 32766, 32767, +/-32768, -32767, -32766, -3, -2, -1

The positive and negative numbers meet in the middle! It seems to make little sense, but if you go far enough in machine language programming, you will discover you can play some interesting tricks with numbers using this scheme and that it truly is the way it ought to be.

The point is this: Any time you have a number larger than 32767 that you wish to use in Integer BASIC, you simply subtract 65536 from it to turn it into its negative counterpart, known technically as its two's complement. Thus,

492ØØ - 65536 = -16336

Magic. And if you want to convert a negative number into its two's complement positive form, add 65536 to it. Since Integer BASIC allows no number larger than 32767, to enter the number 32768 (for example, to set HIMEM: to maximum memory for a 32K system), type:

```
HIMEM: -32767 -1
```

FRAME 9

The truth comes out. The real reason that $CØØ was used as the starting address is so the program could be listed by typing "CØØL". While this is not as clever as 3 DØG for getting back into DOS, it was the cleverest address still available.

FRAME 10

Congratulations. You have succesfully constructed an instrument that may one day displace the famous East-Indian Rubberized Bicyclefish as the weirdest musicmaker of all time. You have only to type CØØG for the fun to begin.

Experiment with the paddles. You will find that not only can you vary the pitch, you can pinch off almost all the sound at some settings -- a real advantage. When you have had as much as you (or someone in the room bigger than you) can stand, press the RESET key to stop the tone. If you want to look at the program again, type CØØL. If you want to make it go again, type CØØG.

If you would like to do some further experimenting, including making the tone kit stereophonic, continue reading. Otherwise, return to the main diskette menu by rebooting the diskette. To reboot from the Monitor type

```
<slot#> CTRL-P
```

To reboot from BASIC type

```
PR# <slot#>
```

THE MULTI-TONE GENIE

Whether or not you wish to continue experimenting, you have now earned the right to invoke the Multi-Tone Genie whenever you want to use the Multi-Tone Kit. The Genie will, with great swiftness and infinite patience, type in all the commands for you and turn on the instrument.

Now that you have assembled the kit for yourself, you can fully appreciate the delicacy and awesome responsibility of the Genie's job. But fear not, the Genie will do it perfectly every time.

READY REFERENCE

Before launching into further discussions, here is a listing of the Multi-Tone program (before modifications):

```
ØCØØ-    A2 ØØ       LDX    #$ØØ
ØCØ2-    2Ø 1E FB    JSR    $FB1E
ØCØ5-    8D 3Ø CØ    STA    $CØ3Ø
ØCØ8-    A2 Ø1       LDX    #$Ø1
ØCØA-    2Ø 1E FB    JSR    $FB1E
ØCØD-    8D 3Ø CØ    STA    $CØ3Ø
ØC1Ø-    4C ØØ ØC    JMP    $ØCØØ
```

You will find that the disk protection scheme keeps you from saving the machine language program on the APPLE HOW TO! diskette. You may save the program on another diskette by

1. Booting up with a standard DOS 3.3 diskette.
2. Entering the Monitor by typing
   CALL -155
3. Making RESET go to the Monitor by typing
   3F2:69 FF 5A
4. Typing the program in manually.

Step four sounds like a lot of work, but you really only have to type this one line:

```
CØØ:A2 ØØ 2Ø 1E FB 8D 3Ø CØ A2 Ø1 2Ø 1E FB 8D 3Ø CØ 4C ØØ ØC
```

This is the same program typed in using the hexadecimal numbers. You may then confirm that the result matches the assembly listing above by typing:

```
CØØL
```

You can then save the results on your diskette by typing:

```
BSAVE MYTONE, A$CØØ, L19
```

The A$CØØ, L19 tells the Disk Operating System that the program is located at the hexadecimal address, $CØØ, and has a Length of 19 bytes. The 19 in this case is in regular old base 1Ø. (You can also use L$13, as 13 in hexadecimal is the same as 19 in base 1Ø.)

After doing that, you may always load the program back in from that diskette by typing

```
BLOAD MYTONE
```

after carrying out the first three steps above to prepare the Monitor. If you wish to save any of the variations of the program outlined below, BSAVE them under another name, being careful to count how many bytes long they are. Please refer to the DOS manual for futher information on the subject of saving and loading machine language programs.

Doing the wrong thing in the wrong place in the Monitor can turn on the disk drive and start writing weird things all over your programs. It is wise to open the door of your disk drive(s) before doing any experimenting. When you are through working in the Monitor, boot the diskette again in case anything in the Disk Operating System has been altered.

This caution should not keep you from experimenting. There are no Monitor commands that are going to hurt the Apple. You will find that even the most experienced machine language programmers open their disk drive doors before trying out new programs; it is so easy to do and it could save you a lot of grief.

MINI-ASSEMBLER COMMANDS

Below are the commands for the Mini-Assembler. Keep in mind that the Mini-Assembler is hidden within Integer BASIC. Thus the Apple must already be in Integer BASIC when you enter the Monitor to work with the Mini-Assembler.

To turn on the Mini-Assembler, type its starting address and a "G" for "Go":

```
$F666G
```

To assemble an instruction into machine code, type:

```
address : 65Ø2 mnemonic <data>
```

for example:

```
$CØØ : LDX #$Ø
```

To assemble subsequent instructions into machine code, type:

space : 65Ø2 menemonic <data>

for example:

:JSR $FB1E

To use a Monitor command from the Mini-Assembler without having to turn the Mini-Assembler off, precede the command with a dollar sign:

$<Monitor command>

for example:

$CØØL

will list 2Ø lines of program starting at $CØØ. Because of this feature, you may not use a dollar sign before an address at the beginning of a line. For example:

CØØ: LDX #$Ø

is correct, while:

$CØØ: LDX #$Ø

will cause the Monitor to receive the CØØ and the L from LDX and proceed to list 2Ø lines starting at $CØØ. Throughout the program, we have been telling you repeatedly that the dollar sign means the number is hexadecimal. Now we tell you that it means "don't use the Mini-Assembler, use the Monitor." Well, it means each, sometimes. At the beginning of a line it means "don't use the Mini-Assembler"; anywhere else in the line it refers to hexadecimal.

The complete list of Monitor commands is in the APPLE II Reference Manual.

To exit from the Mini-Assembler, press the RESET key.

THE LEASE BREAKERS

Once you have tried out the Multi-Tone Kit program, there is an experiment you can attempt that will truly astound both you and your neighbors. Instead of sending the sound to the built-in Apple speaker, try sending it to your stereo!

To do this, we will output the sound to the cassette-out jack on the back of the Apple. You need a patch cord with a mini-plug on the Apple end and a phono plug on the amplifier end. Patch cords can be obtained at any stereo store and will prove useful for many of Apple's music programs too.

The switch for the cassette-out jack is located at address $CØ2Ø. (Like the switch for the Apple's speaker, this switch alternately pulses up and down.) The program currently looks like this:

```
ØCØØ:    A2 ØØ       LDX    #$ØØ
ØCØ2:    2Ø 1E FB    JSR    $FB1E
ØCØ5:    8D 3Ø CØ    STA    $CØ3Ø
ØCØ8:    A2 Ø1       LDX    #$Ø1
ØCØA:    2Ø 1E FB    JSR    $FB1E
ØCØD:    8D 3Ø CØ    STA    $CØ3Ø
ØC1Ø:    4C ØØ ØC    JMP    $ØCØØ
```

We will need to change the commands beginning at $CØ5 and $CØD to STA $CØ2Ø instead of STA $CØ3Ø. To do so, from the Monitor, type F666G which will turn on the Mini-Asssembler and print the ! prompt. Now type

```
CØ5: STA CØ2Ø
CØD: STA CØ2Ø
```

Then, after connecting your amplifier and turning down its volume, type $CØØG

Turn the paddles to the middle and turn up the amplifier to a comfortable (?) setting.

An even more dramatic result occurs when both the amplifier and the built-in Apple speaker are being toggled. To do this, you can STore the Accumulator alternately at $CØ2Ø and $CØ3Ø. To do this type

```
CØ5: STA CØ2Ø
CØD: STA CØ3Ø
```

You should notice a very distinct feeling of stereo separation.

OTHER MADNESS

Abandon sound altogether. Put in some other pairs of switches, such as the color graphics on and off (CØ5Ø and CØ51). Try anything that comes to mind. Just make sure you have your disk drive door open first!

Here is a really wild one: replace the two STA CØ3Ø's with JSR FCAØ's. Then, instead of typing $CØØG, type

```
$CØ5Ø CØ52 CØØG
```

Now, that's good for a full 3Ø seconds of entertainment!

FURTHER EXPLORATION

Machine language offers you the greatest speed and widest range of creativity available on the Apple. If you would like to work more with it, we recommend your using a full assembler, such as EDASM (short for EDitor-ASeMbler) found on the diskette that comes with the APPLESOFT TOOL KIT. A full assembler allows you to insert and delete lines easily, use names for subroutines, and generally make program listings more readable.

Your dealer should also have a stock of books on 6502 machine language. Several good books are available that require very little prior experience on your part (less experience than you have now had). Thank you for sharing this experience with us and good luck with your future programming.

MULTI-TONE KIT GLOSSARY

Accumulator: A register inside the microprocessor. The contents of the Accumulator can be added, subtracted, multiplied, divided and tested for logical conditions, e.g. results = Ø, results < Ø, etc.

Address: A number that designates a location in the computer's memory.

Assembler: A program that converts assembly-language instructions into machine-language instructions.

Assembly language: A computer language made up of simple words, called mnemonics, that are subsequently converted to machine language by an assembler. Assembly-language programs are less difficult to write and understand than programs written directly in machine laguage.

Bit: The smallest piece of information that a computer can hold. A single bit specifies a single value of either "Ø" or "1". Bits grouped together form bytes.

Byte: A basic unit of measure of a computer's memory. A byte comprises eight bits.

Hexadecimal numbering system: A numbering system that uses the ten digits Ø through 9 and the six letters A through F to represent values in base 16. Assembly-language instructions often use hexadecimal notation.

Memory: An area within the computer into which data and programs can be entered for storage and from which they can be retrieved as needed.

Machine language: A computer language that directly controls the lowest level, most fundamental internal operations of the computer.

Microprocessor: The "brain" of the computer. The microprocessor is responsible for executing instructions that control the use of memory and for performing arithmetic and logical operations.

Mnemonic: A label consisting of a short word or group of letters that is easy to remember. Assembly-language instructions are mnemonics.

Monitor: A computer program that allows the user to initiate machine language instructions.

Registers: Locations inside the microprocessor used for temporary storage of data and address information.

X Register: A location inside the microprocessor used for temporary storage of data. The contents of the X register can be tested for logical conditions, e.g. contents = Ø, contents <> Ø, etc.

Y Register: A location inside the microprocessor used for temporary storage of data. The contents of the Y register can be tested for logical conditions, e.g. contents = Ø, contents <> Ø, etc.

INDEX*

A

B

C

D

E

F

* Unless otherwise indicated, all commands refer to RPN Calculator

G

H

I

J

K

L

M

10260 Bandley Drive
Cupertino, California 95014
(408) 996-1010

030-0140-A